REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimates or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1997

3. REPORT TYPE AND DATES COVERED
Technical Report

4. TITLE AND SUBTITLE
Synthesis of ESI Equivalence Class Combinational Circuit Mutants

5. FUNDING NUMBERS
DAAH04-94-G-0280

6. AUTHOR(S)
J. Harlow, F. Brglez

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Collaborative Benchmarking Laboratory
Department of Computer Science
North Carolina State University
Box 7550 NCSU
Raleigh, NC 27695-7550

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park,, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 33616.8-EL

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Despite more than a decade of experience with the use of standardized benchmark circuits, meaningful comparisons of EDA algorithms remain elusive. In this paper, we introduce an entirely new methodology for characterizing the performance of Binary Decision Diagram (BDD) software. Our method involves the synthesis of large equivalence classes of Entropy--Signature Invariant (ESI) circuits, based on a known reference circuit. We demonstrate that such classes induce controllable distributions of BDD algorithm performance, which provide the foundation for statistically significant comparison of different algorithms.

14. SUBJECT TERMS
BDD, entropy, ESI, benchmarking

15. NUMBER IF PAGES
9

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

CBL (Collaborative Benchmarking Laboratory)
Department of Computer Science
Campus Box 7550
North Carolina State University
Raleigh, NC 27695

Synthesis of ESI Equivalence Class Combinational Circuit Mutants

Justin E. Harlow III   Franc Brglez

Technical Report 1997-TR@CBL-07-Harlow
October 1997

If you choose to cite this report, please add the following entry to your bibliography database:

```
@techreport{
1997-TR@CBL-07-Harlow,
author = "J. Harlow and F. Brglez",
title  = "{Synthesis of ESI Equivalence Class Combinational Circuit Mutants}",
institution = "{CBL, CS Dept., NCSU, Box 7550, Raleigh, NC 27695}",
number = "1997-TR@CBL-07-Harlow",
month  = "Oct",
year   = "1997",
note   = "{Also available at http://www.cbl.ncsu.edu/publications}"
}
```

To contact the Collaborative Benchmarking Laboratory via Internet, you may consider:

WWW :	`http://www.cbl.ncsu.edu/`
Anonymous FTP :	`ftp://ftp.cbl.ncsu.edu`
For an auto-reply:	`benchmarks@cbl.ncsu.edu`

To deposit a file at CBL:	`ftp cbl.ncsu.edu`
	`cd /pub/Incoming`
	`put new_benchmark.tar.Z`

Synthesis of ESI Equivalence Class Combinational Circuit Mutants

Justin E. Harlow III[1] Franc Brglez[2]

[2]CBL (Collaborative Benchmarking Lab), Dept. of Comp. Science, Box 7550, NC State U., Raleigh, NC 27695
http://www.cbl.ncsu.edu/

Abstract – *Despite more than a decade of experience with the use of standardized benchmark circuits, meaningful comparisons of EDA algorithms remain elusive. In this paper, we introduce an entirely new methodology for characterizing the performance of Binary Decision Diagram (BDD) software. Our method involves the synthesis of large equivalence classes of Entropy–Signature Invariant (ESI) circuits, based on a known reference circuit. We demonstrate that such classes induce controllable distributions of BDD algorithm performance, which provide the foundation for statistically significant comparison of different algorithms.*

Keywords: *Benchmarking, BDD, entropy, ESI.*

Introduction

The characterization and comparison of Design Automation algorithms has historically been based on the use of widely–accepted sets of benchmark circuits[1, 2]. The use of standardized test cases creates an impression of generality and lack of bias in comparisons, but in fact, there is little generality in this process. The commonly used benchmark circuits do not represent a continuum of complexity, and in general, the benchmark sets have no invariant properties: each circuit is unique in size, function and architecture. Furthermore, since these circuits are widely available and thoroughly studied, it is likely that many current DA tools are, in effect, "tuned" to run well on these circuits. *The authors do not mean to suggest any deception or dishonesty. The mere use of a small sample of specific, unrelated circuits as measures of performance inherently induces bias.*

In most other fields of science, comparisons between theories are routinely made by use of controlled experiments. Acceptance of results by the community is conditioned on demonstration that a theory is repeatable and statistically significant; publication is often denied in cases of minor or incremental advances. The field of Electronic Design Automation, in sharp contrast to such rigor, does not have any process analogous to "clinical trials" as understood in medicine. Incremental or nonrepeatable results routinely appear even in widely–respected conferences and journals, because there is no objective means available to make meaningful comparisons to prior results. It is our long–term goal to introduce a rigorous, scientific, statistically meaningful paradigm to EDA benchmarking.

This paper introduces Entropy–Signature Invariant (ESI) classes of mutant circuits to study the complexity of software packages implementing Reduced, Ordered Binary Decision Diagrams[3] (hereafter referred to as BDDs). In principle, since a BDD is a canonical representation of a *function*, the circuit implementation of that function should have no impact on the complexity of the BDD. However, it is well known that the choice of BDD variable order has a profound impact on the size of the BDD data structure, and that determining an optimal ordering is itself an *np*–complete problem. Developers of BDD packages must therefore employ heuristics to estimate an initial ordering, and often invoke *dynamic reordering* algorithms to refine the initial ordering and keep the complexity of the BDD within bounds. While many studies have been published comparing various BDD packages and algorithms, none have used controlled experimental procedures to show the *sensitivity* of the algorithms to properties of the functions being represented.

The paper is organized into the following sections:
(2) background and motivation;
(3) entropy-invariant classes;
(4) preliminary experiments;
(5) synthesis of ESI mutant classes;
(6) summary of experimental results;
(7) conclusions.

Motivation

Variable ordering for BDD construction has been widely studied, and many elegant algorithms have been published which aim to achieve near–optimal orders at reasonable cost [4, 5, 6]. Nevertheless, a great deal of variability of results can be observed under widely–used BDD packages, even on very simple circuits.

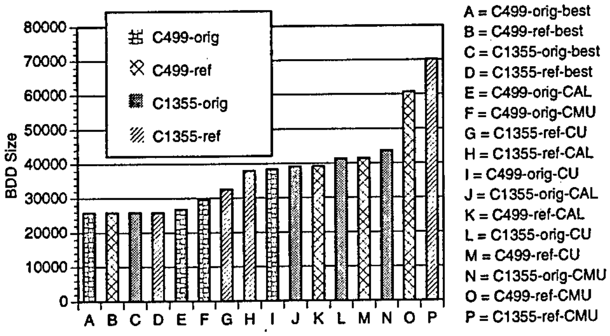


Fig. 1. BDD variability in logically equivalent circuits.

Figure 1 illustrates BDD sizes achieved on the logically equivalent C499 and C1355 combinational benchmark circuits under three BDD packages. For these experiments, we used two structural variants of the benchmark circuits:

(1) Original: The original benchmark circuit was resynthesized using only two–input gates.

(2) Reference: A synthesized version of the original circuit, which has been converted to a canonical bipartite graph

Franc Brglez has been supported by contracts from the Semiconductor Research Corporation (94–DJ–553), SEMATECH (94–DJ–800), and DARPA/ARO (P–3316–EL/DAAH04–94–G–2080).

representation as described in a later section.

In Figure 1, bars A, B, C, and D give the BDD sizes achieved for the two variants of both circuits, with a fixed, near–optimal variable ordering given by Somenzi [7]. As expected, all BDDS were of identical size and were verified to be logically equivalent[1].

The remaining bars in the figure show the BDD sizes achieved using dynamic variable ordering in VIS [8], with the CAL [9], CU [10], and CMU [11] packages. The figure illustrates several important points:

(1) All BDD packages make use of structural analysis to calculate initial variable ordering. The algorithms can easily be led to poor orderings by the particular circuit realization being processed. For instance, the CAL package produced a BDD with 26,790 nodes for the original (E) realization of C499, and 39,166 for the reference (K) version.

(2) Comparisons of the performance of different BDD packages based on isolated benchmarks can be very misleading. For instance, the CMU package got a near-ideal size of 29,606 nodes for the original C499 circuit, but 70,198 nodes for the reference C1355 circuit; the CU package got 32,546 nodes for the C1355 reference, but 41,386 for the C499 original. Depending on which of these logically equivalent circuits is chosen as a benchmark, either package can be said to perform "better."

(3) Each BDD package achieved results close to the known-good ordering in at least one case, but wide variability in performance was observed for all packages under the four function realizations used in the experiment. The worst–case spread was 8,840 nodes for CU, 12,376 for CAL, and 40,592 for CMU.

These observations illustrate the inadequacy of current benchmarking procedures for comparing the performance of function–based algorithms, in particular BDD packages. In this paper, we introduce a new class of test cases which are based on controlled structural variations which preserve the entropy of the reference function. We analyze the properties of this class of circuits and demonstrate their effects on BDD package performance and variability. Through this kind of analysis, we will gain insights into structural factors which affect variability of BDD sizes, leading to methods which will reduce the variability of the BDD size to a range commensurate with the variability in the functions themselves.

Entropy-Invariant Classification

Cheng and Agrawal[12] showed that the *computational work* of a function (thus its complexity) is directly related to its entropy.

For a single–output function, the entropy is given as

$$E(P_1) = P_1 \cdot \log_2\left(\frac{1}{P_1}\right) + (1 - P_1) \cdot \log_2\left(\frac{1}{1 - P_1}\right) \qquad (1)$$

where P_1 is the probability of the output being a "1". It is clear that any two single–output functions whose exhaustive simulations contain the same number of "ones" will have the same entropy, and thus similar complexity.

Figure 2(a) illustrates the concept of entropy invariant mutations, using a simple 3–input multiplexor function $f.00 = ac + bc'$. The string [00100111] represents the output of an exhaustive simulation of the function. Functions $r.01$ through $r.16$ are randomly generated with the same entropy as $f.00$; a simulation, the BDD size and Hamming Distance from $f.00$ are shown for the functions in this entropy–invariant equivalence class. Figure 2(b) illustrates another such class, consisting of mutations of $f.00$ in which the Hamming Distance of any mutant from $f.00$ is exactly 2. Even for such simple functions, it can be seen that a range of BDD sizes is induced by these mutations. Although it is not evident in this trivial example, we will demonstrate that the BDD size range increases as the Hamming distances of the mutants from the reference function increases.

(a) Entropy-invariant Boolean functions in *random* HD class

		HD	BDD
f.00=[00100111]		-	5
r.01	00011101	4	7
r.02	00011110	4	6
r.03	00101011	2	7
r.04	00110011	2	2
r.05	01001011	4	6
r.06	01001110	4	7
r.07	01111000	6	6
r.08	01101001	4	6
r.09	01110010	4	7
r.10	01110100	4	7
r.11	01111000	6	6
r.12	10010101	4	6
r.13	00110110	2	6
r.14	11011000	8	7
r.15	10110001	4	7
r.16	11100001	4	6
----	01234567		

(b) Entropy-invariant Boolean functions in *constant* HD class

		HD	BDD
f.00=[00100111]		-	5
f.01	10000111	2	6
f.02	10100011	2	7
f.03	10100101	2	4
f.04	10100110	2	6
f.05	01000111	2	7
f.06	01100011	2	6
f.07	01100101	2	6
f.08	01100110	2	4
f.09	00010111	2	7
f.10	00001111	2	2
f.11	00110011	2	2
f.12	00110101	2	7
f.13	00110110	2	6
f.14	00101011	2	7
f.15	00101101	2	6
f.16	00101110	2	7
----	01234567		

f.00=ac+bc': reference function
HD: Hamming Distance
BDD: size of BDD (in VIS)

Fig. 2. Two classes of entropy-invariant Boolean functions.

(a) Reference circuit PO-cut

r.13	00110110
r.14	11011000
r.15	10110001
r.16	11100001
----	01234567

(b) Mutant circuit PO-cut

p.13	01010110
p.14	10111000
p.15	11010001
p.16	11100001
----	01234567

Fig. 3. Exhaustive simulation of two 3-input circuits.

For a multi–output function, the entropy is given as

$$E(P) = \sum_{i=1}^{2^m} P_i \cdot \log_2\left(\frac{1}{P_i}\right) \qquad (2)$$

where P_i is the probability of primary output vector i under an exhaustive simulation of the function, and m is the number of primary outputs. In a manner analogous to single–output functions, multi–output functions have the same entropy if their exhaustive simulations have the same number of instances of each vector occurring at the outputs.

Entropy Signature. Consider a combinational circuit whose signal values are probed on k wires. We refer to the wires being probed as a *cut.* For example, the results of the two simulations in Figure 3 correspond to two 3-input 4-output combinational circuits. The four primary outputs can be considered as a PO-cut, and its entropy, under exhaustive simulation, can be evaluated exactly using Equation 2. We refer to the entropy of a cut, whether obtained via

[1] For all experiments described in this paper, the VIS 1.1 system[8] was used. An Appendix describes the test conditions and conventions.

exhaustive simulation or approximated, as an *entropy signature* of the cut. Notably, the entropy signature for either of the responses in Figure 3 is 1.5613, thus we say that both responses correspond to a pair of Entropy Signature Invariant (ESI) circuits.

Entropy–Invariant Mutations. We define a *PO Cut* of a multi–output circuit as a cut consisting of the primary outputs of the circuit. Figure 3(a) represents a simulation of the PO cut of a hypothetical 3–input, 4–output function. Figure 3(b) is an entropy–invariant mutation of this circuit. Note that this mutation was achieved by simply exchanging PO vectors (columns in the figure) 1 and 2. In general, any such pairwise vector exchange (or indeed, any $2k$–wise exchange) is entropy–invariant, since only the order of the vectors changes, not their probability. This suggests that, in principle, large classes of entropy-invariant functions can be easily synthesized from a small simulation of a reference function by randomly exchanging pairs of vectors in the PO cut. Later, we will extend this notion to two–vector exchanges in any cut of a canonical form of the circuit.

Size of a Mutation. We define the *size* of such a circuit mutation as the Hamming distance between the reference and mutated functions. In Figure 3, the size of the mutation is 3. While the size of a k–output mutation can range from 1 to k, the Hamming distance between any two single–output functions in the ensemble is either 0 or 2, regardless of the choice of input vector pairs. Thus an entropy–invariant mutation of a multi–output function induces only very minor changes in the constituent single–output functions. Operations of this type are used in the remainder of the paper to generate large ESI circuit classes for characterization of BDD packages.

Properties. The essential properties of each ESI mutant class are the following:

(1) Each circuit in the ESI mutant class has the same signature through the PO-cut.

(2) The size of a perturbation across a PO-cut of size k may vary from 0 to k. Perturbations are performed pairwise in order to preserve the entropy of the cut. However, the size of any perburbation of each function in the cut is bounded by the number of pairs of minterms that are used to induce the perturbation; thus the perturbation size may be at most 2, 4, 6, etc. if we use 1, 2, 3, pairs of minterms respectively.

(3) It is not necessary to always perturb the PO-cut to induce an ESI mutant circuit behavior. The canonical bidirected graph form, which we introduce later in the paper, can serve the same purpose. The same principles apply to perturbation of any cut defined in the canonical form.

Preliminary Experiments

In principle, a "small" change in a function should result in a correspondingly small change in its BDD. We now illustrate the effect of small entropy–invariant mutations on the size of BDDs, to demonstrate that ordering algorithms can be extremely sensitive to these perturbations.

Figure 4(a) shows the mean BDD size for classes of 200 randomly generated functions. Each class contains 200 unique functions with a constant entropy, one output, and a number of inputs ranging from 5 to 11. As expected, BDD size is nearly exponential in entropy for any particular function size. Figure 4(b) shows the detailed distribution for one class, the 11–input functions with E=1.0. The Hamming Distance of this set of randomly–generated functions ranged from 955 to 1115 from a member of the class whose BDD size is the median of the distribution. Figure 4(c) shows the distribution of a different class of 200 circuits, which was created not randomly, but by inducing entropy–invariant mutations as described in the previous Section. The median–size random circuit of the class of Figure 4(b) was again used as a reference circuit, and mutations were performed as described earlier. The narrower distribution of BDD sizes confirms the intuition that BDD size variability should decrease for functions "closer" to the reference circuit.

A common objection to the use of randomly generated functions in benchmarking is that they are in some sense not representative of realistic circuits. We now show how entropy–invariant mutations of a realistic circuit can overcome this objection. We usethe modified carry chain circuit of Figure 5(a) as a readily–scalable example of a common circuit; the circuit can be cascaded to any number of inputs, terminated by an XOR gate with a single additional input, e in Figure 5(a), to assure an exact entropy of 1.0. For any size, the exact optimal variable ordering of this circuit is known. The upper line of Figure 4(d) represents the distribution of BDD sizes for the random functions of E=1.0, for 5 through 11 inputs, from Figure 4(a). The lower line shows a similar distribution for entropy–invariant mutations (Hamming distance 2) of the example circuit over the same range of input sizes. Note that at $n = 11$, an anomaly appears for the realistic circuits. Rather than increasing monotonically as expected, the BDD package appears to have found a much better variable order for the mutants at $n = 11$ than for smaller values of n; however, note that this was not the case for the reference circuit.

Figure 4(e) shows the lower line of Figure 4(d) on an expanded scale, along with distributions for the same mutants with a fixed, optimal variable order. The bottom set of points of Figure 4(e) represent the reference circuit alone, under the optimal variable order. It is evident that the ordering algorithms do not produce near–optimal orderings for any of the mutants. Even with an optimal order as a starting point, the dynamic variable ordering algorithms apparently result in worse orderings than they start with. Finally, none of the mutants is as small as the reference circuit.

Figure 4(f) shows the complete distribution for the mutant set at $n = 11$, with the reference function shown to illustrate the anomalous behavior observed above. With this set of experiments, we have demonstrated that on some small, realistic functions, even under known–optimal variable ordering, current BDD variable ordering algorithms can produce surprisingly nonoptimal results. Next, we extend the work to larger circuits and show how the accepted benchmarking methodology for BDD algorithms can be misleading under controlled experimental conditions.

ESI Mutant Class Synthesis

We now present a methodology for scaling these results to larger functions, and demonstrate that the demonstrated properties of entropy–invariant mutant classes are observed on the larger functions.

Bipartite Canonical Form. A graph-based model of a netlist is not effective for the problems we consider. On the other hand, a model of a netlist as a directed hypergraph is not unique. We use the notion of cell level, levels of net pins, and *netspan*[2]. The *canonical form of a bipartite directed graph*, a multi-level graph structure of alternating sets of net nodes and cell nodes, is a simple transformation of the underlying netlist: levels of some of its pins are redefined, and a new type of cell node, a *feedthrough cell* is introduced.

The salient property of this form is that the netspan of all edges (or wires) in this graph is well-defined: edges (wires) connecting net nodes and cell nodes have netspan = 1, edges (wires) connecting cell nodes and net nodes have netspan = 0.

[2] For each net, netspan = $p_{max} - p_{min}$, where the two numbers denote the maximum and the minimum pin level of the net

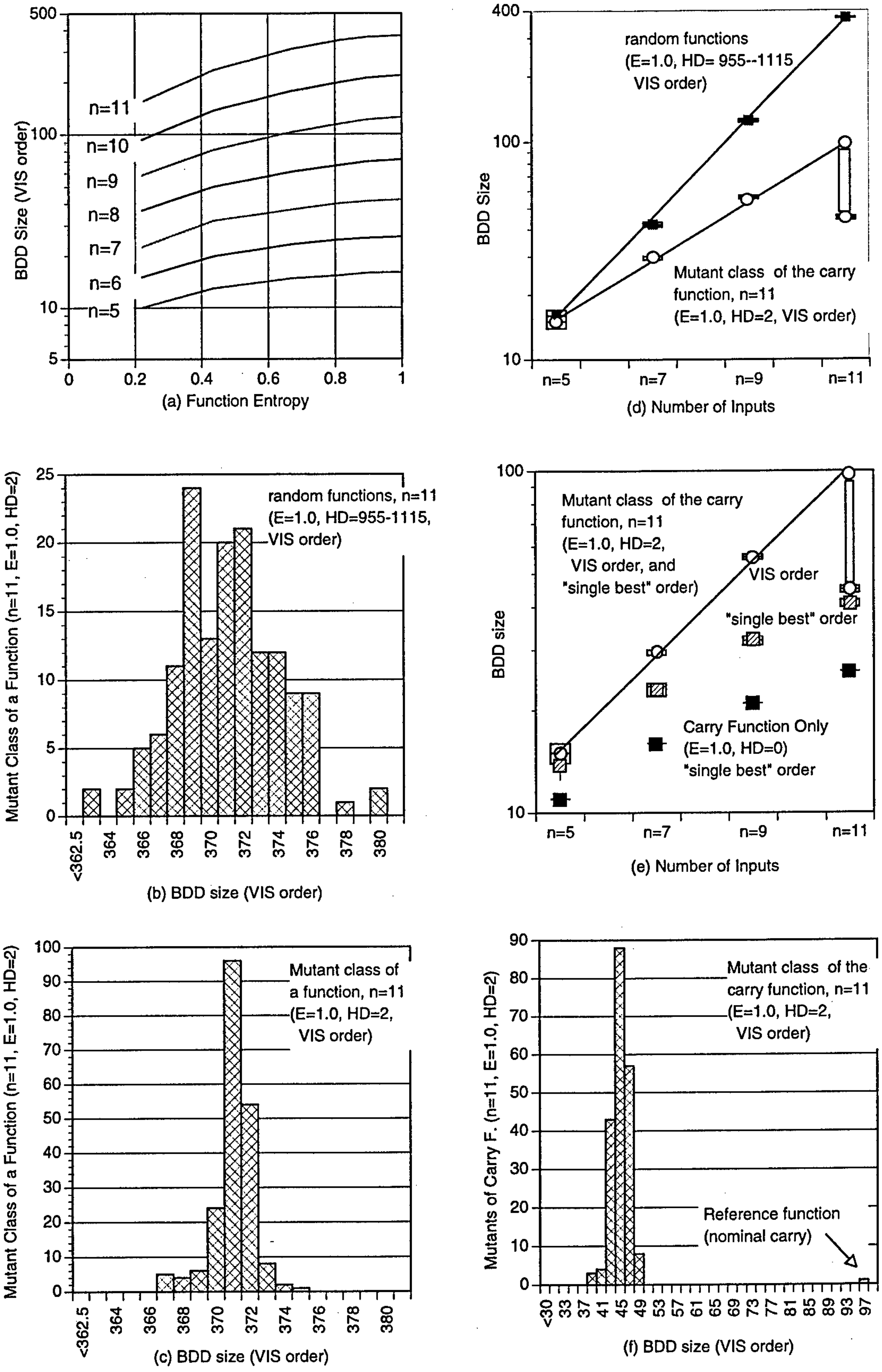


Fig. 4. Preliminary BDD experiments with entropy-invariant Boolean functions.

Interchangeably, we may refer to these edges as *wires* or *two-pin nets* [13]. The canonical form provides for simple definition of a *cut*, in addition to the PO-cut defined earlier. We define the cut by probing, at a chosen level k, *all* wires driven by logic and feedthrough nodes *before* the wires reach the respective net node, *i.e.* all wires whose netspan = 0. These are the same wires onto which we induce entropy-signature invariant perturbations, as explained earlier.

We illustrate an example of a circuit schematic with its canonical form and pairwise entropy signature in Figure 5(a-b). The circuit is the modified carry chain used in the previous discussion. In addition, Figures 5(c-d) have been designed to illustrate the process of logic perturbation that leads to the the synthesis of the mutant equivalence class.

Mutant Synthesis. Consider the reference circuit in Figure 5(b). The circuit is annotated, at each level cut, with the results of simulation of two randomly chosen input patterns, [11110/00100]. A summary of this simulation, with a few additonal patterns, is shown in the accompanying table. Specifically, the entries in the table summarize the number of transitions observed in a specific level cut, including the PI-cut and PO-cut. For example, at level L1, we observe a 4-wire cut, with three wires having transitions, which propagate until observed at the primary output. Notably, the 3 transitions at level 1 correspond to the size of a cut perturbation with a Hamming distance of 3.

Given the 4-pattern pair signature in Figure 5(b), we have many choices to synthesize a mutant in the ESI class. Figure 5(c) illustrates the choices of perturbations which could be induced when decoding the simulated pattern [11110/00100]. Such a mutation in a canonical circuit netlist can be implemented as shown in 5(d). At each mutation site in a chosen cut, a 2-input XOR gate is inserted, with its other input driven by a signal z generated by decoding the two vectors. To maintain entropy-invariance, we must perturb all wires with observed transitions *at same level cut.* For example, on level 1, we introduce three perturbations, while on level 3 we introduce 2 perturbations. Interestingly, introducing perturbations in an even number of levels not only maintains the entropy but also maintains *logic* invariance of the function. Clearly, in a circuit with q levels, and at least 1 perturbation at each level, the number of mutants that can be generated is proportional to $q!$.

Note the following important features of this method:

(1) Using different vector pairs, or different cuts of a canonical circuit, arbitrarily large classes of entropy–invariant mutant circuits can be generated cheaply. The method scales to large circuits, and requires only simulation of a few randomly selected vectors to generate large classes of mutant circuits.

(2) Although the selection of vectors and cuts is deliberately random, the resulting mutant circuits are *not* random. All mutants thus generated are closely related to the reference circuit, and can be expected to induce a controlled distribution of BDD sizes.

Although this paper deals only with the first such class, we are studying the generation and properties of ESI mutants in three subclasses:

- ESI mutants based on perturbations of the PO cut.
- ESI mutants based on a selected "internal" cut.
- ESI mutants which are also logically equivalent to the reference circuit, produced by perturbing an even number of selected cuts.

Experiments

We chose the C499 and C1355 benchmark circuits [1] as reference circuits for the following experiments. These logically–identical circuits have 41 inputs and 32 outputs; the reference circuits consist of 206 (C499) and 518 (C1355) 2–input gate equivalents. In their respective graph–canonical forms, the sizes become 694 and 1542 due to the addition of feedthroughs.

We used the VIS 1.1 [8] software from the University of California at Berkeley as a platform for the experiments. Some of the conventions and procedures used are described in the Appendix.

Classes of 100 ESI mutants for C499 and C1355 were generated by perturbing omly the PO cut. For each class, BDDs were built for each mutant, using VIS 1.1 and each of its three available BDD packages. Except in Experiment 1, the software was allowed to compute a static variable ordering using its built–in algorithms, with no "hints" supplied. In all cases, dynamic variable ordering was enabled, using the *sifting* method.

The experiments were as follows:

1. In order to verify that the mutant classes behaved as the small circuits in our preliminary experiments (Figure 4), we built BDDs for the mutant classes using a fixed, near–optimal static order given by Somenzi [7]. Figure 6(a) presents the mutant sizes for C499 under the CU package using this ordering; the mean was 26,649 and the standard deviation was 152 nodes. Figure 6(b) shows the differences between the sizes for C499 and C1355. As expected, the difference of the distributions was quite small, reflecting the logical "proximity" of the functions in each class to the reference circuit.
2. We repeated the first experiment, this time allowing each of three BDD packages to compute a static variable ordering and to dynamically reorder the variables, using the sifting method [4]. Figure 7 summarizes the results of this experiment. Panels (a) and (d) show the distribution of BDD sizes achieved by the CU (University of Colorado, Boulder) BDD package for C499 and C1355 respectively[3]. The remaining panels show the distribution of *differences* between BDD sizes for each specific mutant circuit relative to the the CU results. Panels (b) and (e) show the CAL package (University of Califormia, Berkeley), while panels (c) and (f) show the CMU (Carnegie–Mellon University) results. We use this differential presentation to highlight detailed differences among the packages, which would be difficult to discern from raw data.
3. Both experiments were repeated several times using unique mutant classes of 100 circuits generated from the same reference circuits by the same methods, with results statistically indistinguishable from those shown here.

Summary of Results

The experiments described above result in the following observations:

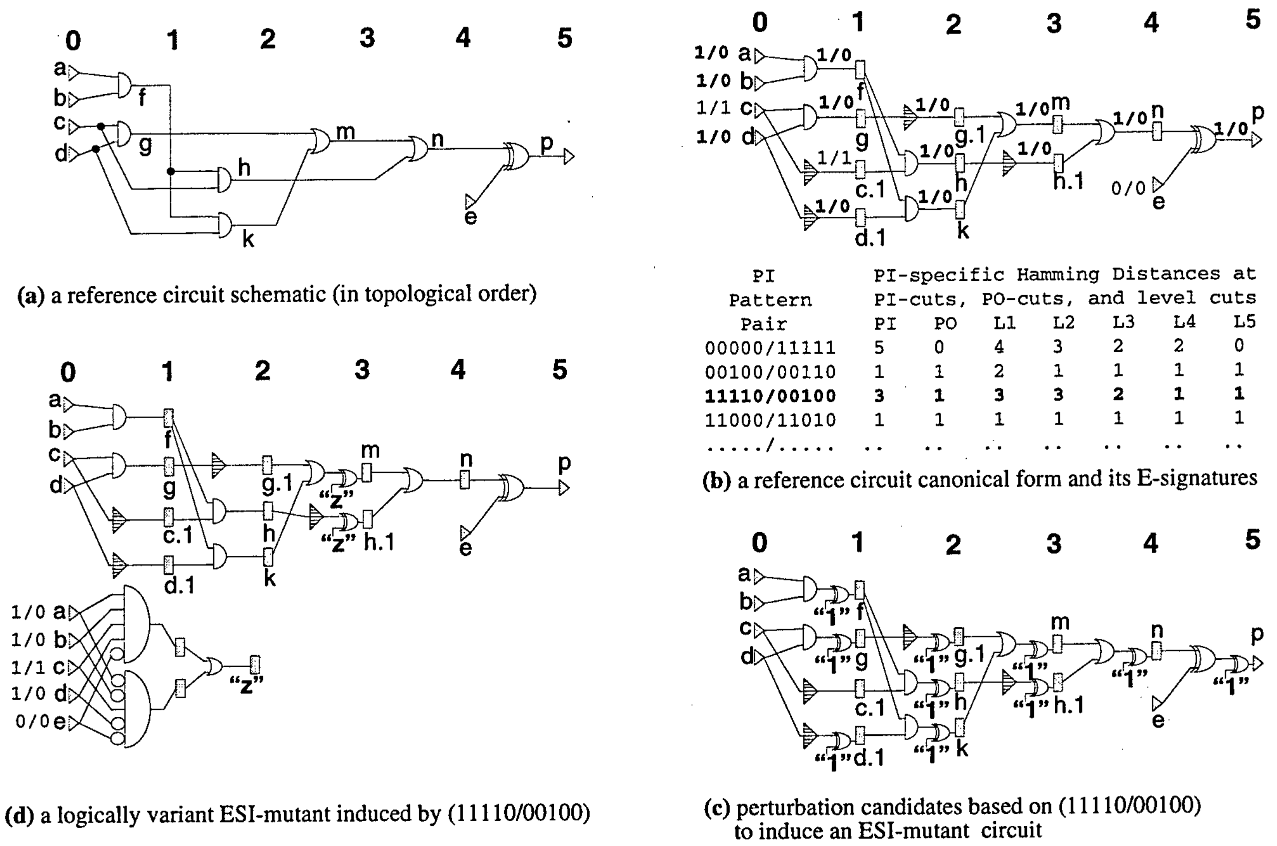
(1) Overall, the CU package achieved the most uniform results. The average BDD size over the mutant classes was 41,644 nodes for the C499 and 38,165 for the C1355. The precision of the estimate of these means was 2.69% for the C499 and 4.8% for the C1355, at the 95% confidence level.

(2) The CAL package got smaller BDDs than the CU package for a majority of the C499 mutants, but larger for most of the C1355 mutants.

(3) The CMU package got larger BDDs for both circuits in most cases.

(4) For the "best" variable ordering, the distribution of BDD sizes was very narrow over the mutant classes, in contrast to the distributions of Figure 7 for computed orderings.

[3] We chose the CU results as a reference point because its distribution had the smallest variance of the three packages tested.



(a) a reference circuit schematic (in topological order)

PI Pattern Pair	PI-specific Hamming Distances at PI-cuts, PO-cuts, and level cuts						
	PI	PO	L1	L2	L3	L4	L5
00000/11111	5	0	4	3	2	2	0
00100/00110	1	1	2	1	1	1	1
11110/00100	**3**	**1**	**3**	**3**	**2**	**1**	**1**
11000/11010	1	1	1	1	1	1	1
...../.....	..	..	..	..	..	..	..

(b) a reference circuit canonical form and its E-signatures

(d) a logically variant ESI-mutant induced by (11110/00100)

(c) perturbation candidates based on (11110/00100) to induce an ESI-mutant circuit

Fig. 5. Circuit schematic, canonical form and ESI mutant generation.

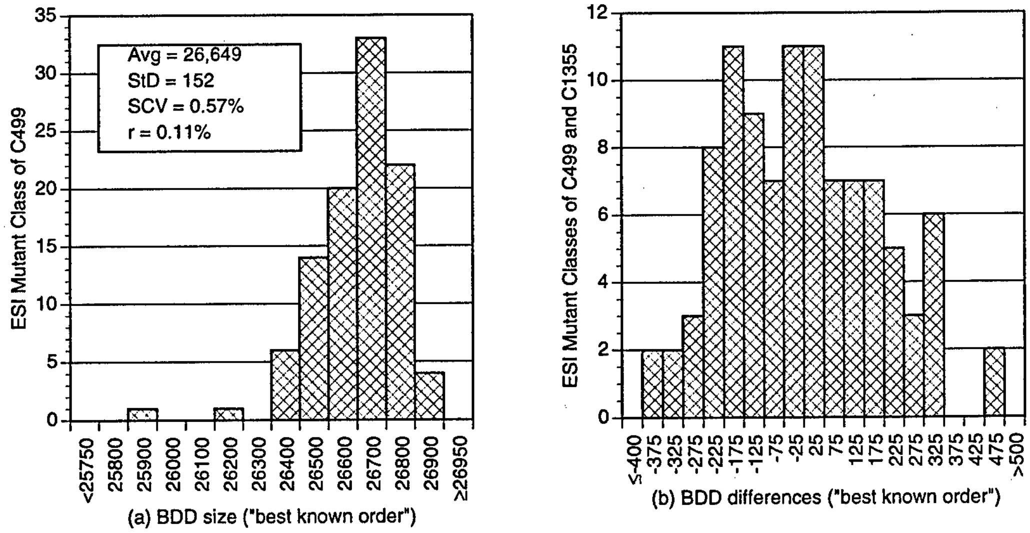
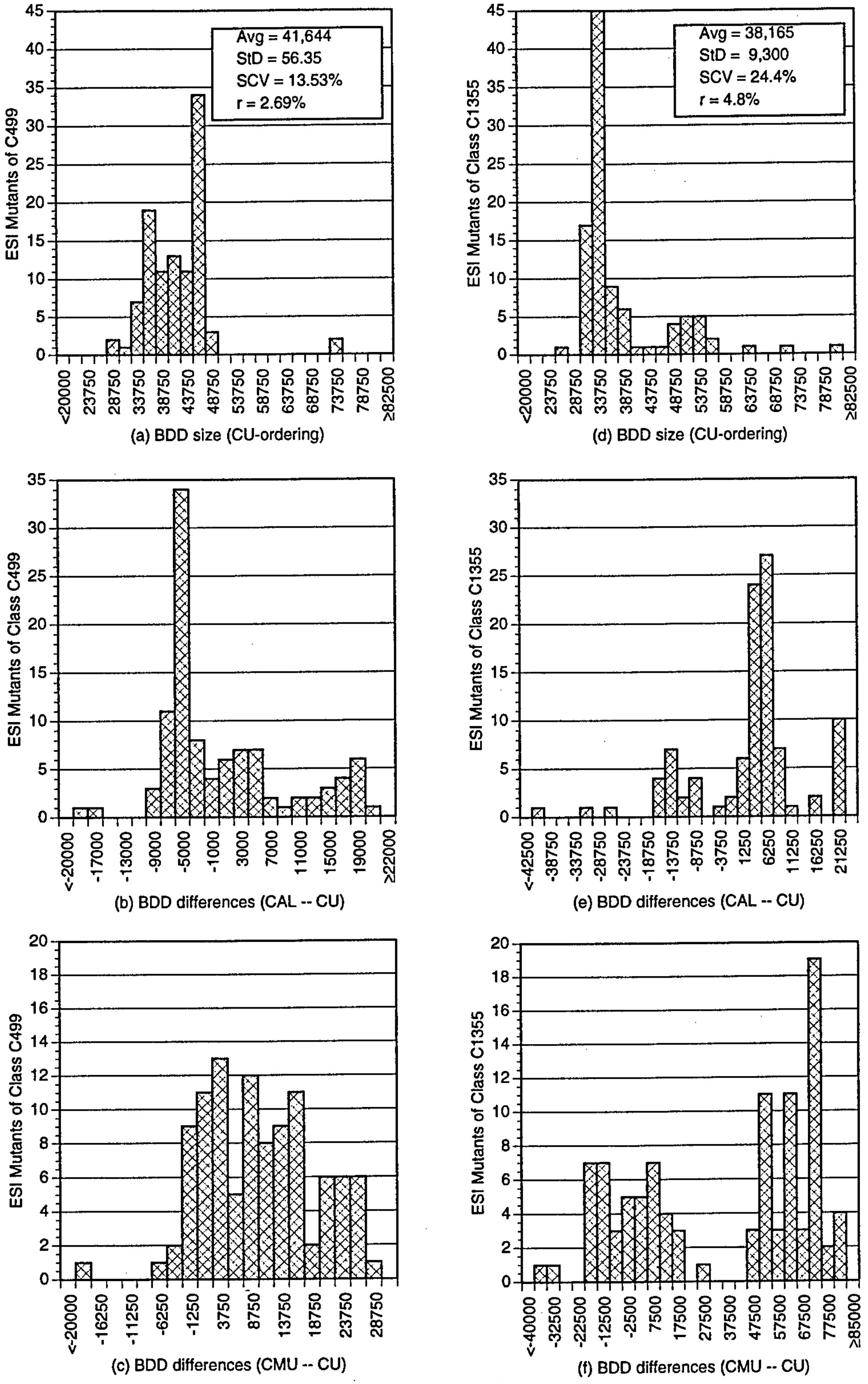


Fig. 6. Best order mutant difference (C1355 – C499).



(a) BDD size (CU-ordering)

(d) BDD size (CU-ordering)

(b) BDD differences (CAL -- CU)

(e) BDD differences (CAL -- CU)

(c) BDD differences (CMU -- CU)

(f) BDD differences (CMU -- CU)

Fig. 7. Absolute and relative BDD histograms with three algorithms for C499 and C1355 PO-cut mutant class.

For the CU package, the mean for all mutants was 26,649 with the best ordering, and a precision of the mean estimate of 0.11%. The contrast between these two distributions is a strong indication of how much improvement is still possible, in principle.

(5) Most importantly, the results presented are *significant* and *repeatable*. This is of critical importance for benchmarking and comparison of results. For instance, the means of the CU and CMU distributions for C499 are, respectively, 41,644 and 51,922, with precision of the estimates of 3.1% and 2.6% respectively, at the 95% confidence level. Thus an assertion such as, "The CU package achieves smaller BDDs than the CMU package" can be made in a statistically significant sense for the first time.

Conclusions

We demonstrated that current BDD packages may exhibit unpredictable behavior under topological variations of the same circuit, thus casting doubt on the significance of many of the published results on these systems. We then introduced the notion of entropy invariance as a way of generating sets of closely related functions, and demonstrated that BDD sizes behave as random variables over classes of these functions. In experiments with both random and "realistic" mutant functions, we showed that algorithm performance was often well-behaved for "nearby" functions, but discontinuous for larger or more "remote" functions.

We then extended these concepts and defined the class of Entropy Signature Invariant circuit mutations, demonstrating two methods for generating large sets of circuits with the entropy property held invariant. We showed that ESI mutant classes, generated from a reference circuit by a simple, scalable methodology, cause BDD variable sizes for the classes to behave as random variables. Using this technique, we illustrate, for the first time, differences between widely-used BDD packages, which are statistically significant and easily repeatable. We intend to apply this methodology to explore the sensitivity of specific ordering algorithms to specific types of circuit perturbations.

It is clear from this work that previous methods of evaluating the performance of BDD algorithms are inadequate. We have defined new methods which will lead to deeper understanding of the sensitivity of algorithms to function properties, and provide a foundation for the measurement and refinement of future algorithms.

Acknowledgements

We acknowledge the cooperation of many researchers who contributed to this work. We owe a special debt to the VIS Group at Berkeley, who provided an excellent platform for our experiments, and superb software support. Fabio Somenzi deserves special mention for the use of his DDCAL and CUDD software, for his collection of BDD orders, and for the many fine points of BDD lore that he clarified for us in the course of the study. Debabrata Ghosh provided software for generating the graph–canonical forms and random functions, as well as answering innumerable questions. Hemang Lavana straightened out many software and system issues for us, and Nevin Kapur provided invaluable assistance in data reduction. The latter three gentlemen are to be commended for their invariable good cheer and willingness to help out. Last, but assuredly not least, Lucy Ringland assisted with proofreading, systems support, and generally keeping the Lab running.

Appendix: Experimental Procedures

Throughout this study, we used the VIS 1.1 software [8], developed and supported by the University of California at Berkeley. VIS provides a uniform application interface and scripting language for (among many other things) building and manipulating BDDs. The following points should be noted:

(1) BDD node counts were based on the VIS–generated count of MDD (multiterminal decision diagram) nodes. These structures use a compact representation which does not store complement arcs, and maintains a single terminal node for each MDD. The "traditional" method of counting (often seen in textbooks) includes both the "0" and "1" terminal nodes, but omits the nodes dedicated to input variables, resulting in slightly different node counts for small circuits. In general, the MDD count is larger than the traditional count by the number of primary input variables, an insignificant difference for reasonable sized circuits.

(2) For all experiments involving random numbers, we used a 48–bit initial seed value derived from dynamic system properties. These initial seeds were saved, and can be used in the future to regenerate the mutant classes and repeat the experiments exactly if need be.

(3) All experiments were performed on circuits encoded in the BLIF language, and were driven by automatic scripts. Many PERL 5 programs [14] were developed to automate the processes and reduce the data.

(4) When a fixed variable order was required for an experiment, we used those archived at the University of Colorado, Boulder [7]. However, on the Carry circuit examples, we provided the obvious best order ourselves.

References

[1] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 695–698, June 1985.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *IEEE 1989 International Symposium on Circuits and Systems – ISCAS89*, pages 1924–1934, May 1989.

[3] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Computers*, C–35(8):677–691, August 1986.

[4] Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Proceedings of the International Conference on Computer Aided Design*, pages 42–47, November 1993.

[5] M. R. Mercer, R. Kapur, and D. E. Ross. Functional approaches to generating orderings for efficient symbolic representations. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pages 624–627, June 1992.

[6] Shipra Panda, Fabio Somenzi, and Bernard F. Plessier. Symmetry detection and dynamic variable ordering of decision diagrams. In *Proceedings of the International Conference on Computer Aided Design*, pages 628–631, November 1994.

[7] Fabio Somenzi. A collection of bdd variable orders. available from `ftp://vlsi.colorado.edu/pub/orders.tar.gz`, 1997.

[8] The VIS Group. VIS: A system for verification and synthesis. In R. Alur and T. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification*, number 1102 in Lecture Notes in Computer Science, pages 428–432, New Brunswick, NJ, July 1996. Springer.

[9] Pranav Ashar and Matthew Cheong. Efficient breadth–first manipulation of binary decision diagrams. In *Proceedings of the International Conference on Computer Aided Design*, pages 622–627, November 1994.

[10] Fabio Somenzi *et al.* Colorado University Decision Diagram package (CUDD), release 2.1.2, 1997. Available from `ftp://vlsi.colorado.edu/pub/cudd-2.1.2.tar.gz`.

[11] David E. Long. CMU BDD package, 1993. Available from `http://emc.cmu.edu/pub/bdd/bddlib.tar.Z`.

[12] Kwang-Ting Cheng and Vishwani D. Agrawal. An entropy measure for the complexity of multi–output Boolean functions. In *Proceedings of the 27th ACMIEEE Design Automation Conference*, pages 302–305, 1990.

[13] D. Ghosh, N. Kapur, J. Harlow, and F. Brglez. Synthesis of Signature-Invariant Equivalence Class Circuit Mutants and Ap-

plications to Benchmarking. Technical Report 1997-TR@CBL-01-Ghosh, CBL, CS Dept., NCSU, Box 7550, Raleigh, NC 27695, February 1997. This report is available as a postscript file via http://www.cbl.ncsu.edu/publications.

[14] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming PERL*. O'Reilly & Associates, Inc., 2nd edition, 1996.